MEMORANDUM
RM-5433-PR
OCTOBER 1967

AN APPRAISAL OF SOME SHORTEST PATH ALGORITHMS

S. E. Dreyfus

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

40

MEMORANDUM
RM-5433-PR
OCTOBER 1967

AN APPRAISAL OF SOME SHORTEST PATH ALGORITHMS

S. E. Dreyfus

This research is supported by the United States Air Force under Project RAND—Contract No. F44620-67-C-0045—monitored by the Directorate of Operational Requirements and Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

The RAND Corporation
1700 MAIN ST • SANTA MONICA • CALIFORNIA • 90406

PREFACE

This Memorandum treats five different problems involving the determination of the shortest path through a discrete network. Previous important results are reviewed, and misleading procedures are identified and (in some cases) modified. Conclusions are drawn and recommendations are made concerning efficient algorithms.

This work forms a part of RAND's continuing interest in problems and techniques in optimization theory.

The author, a member of the Industrial Engineering faculty at the University of California, Berkeley, is a consultant to the Computer Sciences Department of The RAND Corporation.

SUMMARY

This Memorandum treats five discrete shortest-path problems: 1) determining the shortest path between two specified nodes of a network; 2) determining the shortest paths between all pairs of nodes of a network; 3) determining the second, third, etc. shortest path; 4) determining of the fastest path through a network with travel times depending on the departure time; and 5) finding the shortest path between specified endpoints that passes through specified intermediate nodes. Existing good algorithms are identified while some others are modified to yield efficient procedures. Also certain misrepresentations and errors in the literature are demonstrated.

CONTENTS

I. INTRODUCTION

In the never-ending search for good algorithms for various discrete shortest-path problems, some authors have apparently overlooked or failed to appreciate previous results. Consequently, certain recently reported procedures are inferior to older ones. Also, occasionally, inefficient algorithms are adapted to new, generalized, problems where more appropriate modifiable methods already exist. Finally, the literature contains some erroneous procedures. In this Memorandum, we shall briefly consider various versions of discrete path problems in the light of known results and some original ideas, and summarize our conclusions and recommendations.

The reader should realize that our observations are by no means definitive or final. Despite this fact, it is hoped that our somewhat skeptical survey of current literature will put the interested reader on guard and perhaps save him, or his digital computer, considerable time and trouble. Since our objective is more to alert than to resolve conclusively, this Memorandum is informal and, at times, cryptic. We hope that even our most laconic remarks will become insighcful for any reader deeply involved with the particular procedure.

II. THE SHORTEST PATH BETWEEN A SPECIFIED PAIR OF NODES

Given a set of N nodes, numbered arbitrarily from 1 to N, and the NxN matrix D, not necessarily symmetric, whose element d_{ij} represents the length of the directed arc connecting node i to node j, find the path connecting node 1 and node N of shortest length. We assume initially that $d_{ii} = 0$ and $d_{ij} \geq 0$. If there is no arc directed from node i to node j, then $d_{ij} = \infty$, or, for purposes of digital computation, d_{ij} is taken large.

It appears that the computationally most efficient procedure was described first by Dijkstra [1] in 1959,* and in 1960 by Whiting and Hillier.† The algorithm assigns tentative labels, which are upper bounds on the shortest distance from node 1, to all nodes; after repeating the fundamental iterative step described below exactly one time for each node, the tentative node labels are all permanent and represent actual shortest distances. Initially, node 1 is labelled with the permanent value 0, and

*I am indebted to J. D. Murchland for pointing out this reference to me. I would appreciate being told of any even earlier reference for this algorithm or any other one appearing in this Memorandum.

†See Ref. 2, last paragraph beginning on p. 39.

all other nodes tentatively are labelled ∞. Then, one by one, each node label except that at node 1 is compared with the sum of the label of node 1 (i.e., 0) and the direct distance from node 1 to the node in question. The smaller of the two numbers is the new tentative label. Next, the smallest of the N-1 tentative labels is determined and declared permanent. Suppose that node k is the one permanently labelled. Then, one at a time, each of the remaining N-2 tentative node labels is compared to the sum of the label just assigned permanently to node k and the direct distance from node k to the node under consideration. The smaller of the two numbers becomes the tentative label. The minimum of the N-2 tentative labels is determined, declared permanent, and made the basis of another modification of the remaining tentative labels of the type described above. When, after at most N-1 executions of the fundamental iterative step, node N is permanently labelled, the procedure terminates. (If the shortest paths from node 1 to all other nodes are desired, the fundamental iterative step must be executed exactly N-1 times.) The optimal paths can easily be reconstructed if an optimal policy table (in this case, a table indicating the node from which each permanently labelled node was labelled) is recorded. Alternatively, no policy table need be constructed, since it can always be determined from the final node labels by determining which nodes have labels that differ by exactly the length of the connecting arc.

This algorithm requires $\frac{N(N-1)}{2}$ additions and N(N-1) comparisons to determine all node labels. ($\frac{N(N-1)}{2}$ additions and comparisons to compute tentative node labels during all steps and another $\frac{N(N-1)}{2}$ comparisons to pick out the minimum one during all steps.) All steps are naturally and easily programmed except that of recognizing and acting on the basis of which nodes are permanently labelled and which are tentatively labelled. Some computational experimentation* indicates that an efficient way to treat this consideration is to attach to each node an index number which is changed from 0 to 1 (say) when a node label becomes permanent. When branching out from a just-permanently-labelled node, the index of the destination node is consulted as each outgoing arc is considered. If the index is zero, the temporary label of the destination is reduced, if appropriate. At the same time, memory cells containing the smallest temporary label encountered thus

*R. L. Mobley very ably programmed this and all other computational experiments mentioned herein.

far during the branching and the associated destination node are modified if appropriate. This programming device requires $(N-1)^2$ comparisons to consult the indices. Hence, a total of about $N^2/2$ additions and $2N^2$ comparisons are needed, averaging out to about $3N^2/2$ additions and comparisons.

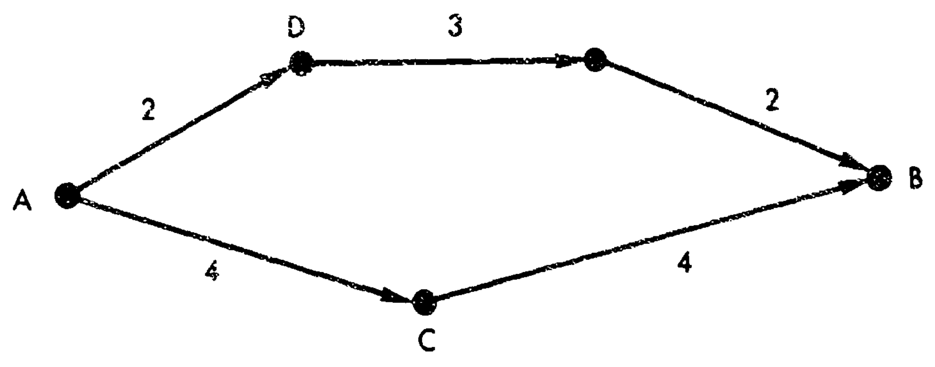
The first procedure described in Ref. 2, and also derived independently by Dantzig [3], results in less additions and comparisons than the above algorithm but involves ordering, from shortest to longest, the arcs out of each node, and cancellations in such lists. Since merely ordering N-1 numbers involves on the order of $N \log_2 N$ comparisons, and this must be done at N-1 nodes, it appears that merely the preparation of the data eliminates this procedure from contention in all but special situations.

Pollack and Weibenson* describe and credit to Minty a systematic and easily programmed permanent label-setting precursor of the above methods. The method, perhaps the first permanent label setting procedure, is probably due originally to Ford and Fulkerson [5] who developed the algorithm for a more general flow problem, of which the shortest-path problem is a special case requiring fastest

*See Ref. 4, p. 225.

flow of one item. In Routine 1 on p. 422 of Ref. 5, if the times t_{ij} are distances and the capacities c_{ij} are all arbitrarily set equal to 1, the node number π_i, when i is labelled, is the shortest distance from P_o to i. (The actual label of node i is [the node previous to i in the shortest path to i, 1] and is irrelevant.) By increasing the π's associated with unlabelled nodes by an amount chosen so that one more node can be labelled, rather than increasing them at each iteration by one, the "Minty algorithm" is obtained. The method requires approximately $N^3/6$ additions and comparisons for solving the shortest-path problem and hence is not recommended.

Some authors have proposed simultaneously fanning out from both endpoints as a means of reducing computation. This advice is sometimes accompanied by the false assertion that when, for the first time, some node is permanently labelled in both fans, the optimal path is determined and goes through that node [6,7]. In the problem depicted below, if the Dijkstra scheme is first used to permanently label the node nearest A reachable from A, then the node nearest B from which B can be reached, then the second closest to A, etc., the node C is permanently-labelled both "out from A" and "into B" after two applications of the procedure at each end, yet ACB is not the shortest path.



When the correct stopping procedure given cryptically by Nicholson [8] and explained clearly by Murchland [9] is used, the least upper bound on the computation required by a two-ended procedure exceeds that for the one-ended Dijkstra algorithm. This is because, as more nodes become permanently labelled in the Dijkstra procedure, a decreasing number of additions and comparisons are needed to modify all tentative labels. As a result, determining the first $\frac{N}{2}$ permanent labels requires roughly $\frac{3}{5}$ths* the

*This number is empirical. Theory based on addition and comparison count predicts about 2/3rds, but machine-dependent and programming-language-dependent details such as how and when various address modifications are computed can cause significant variations in actual computation times. Clearly the first half of a problem requires more than half the work of complete solution, which is the important point.

work of the complete solution. If the Nicholson stopping condition is satisfied considerably before $\frac{N}{2}$ nodes have been permanently labelled from each terminus, a savings may accrue; but, in a case where close to all N nodes must be permanently labelled from either one end or the other, the two-ended procedure will prove inefficient.

All the above methods require that all elements of D be non-negative. A related problem assumes that some d_{ij} are negative, but that the sum of the d_{ij} around any loop is positive. (The problem has no solution if negative loops exist and are allowed to be included in paths. The problem is very difficult and no satisfactory algorithms are known if negative loops exist and the restriction is made that no node can be visited more than once.) While better methods for this problem probably are forthcoming, the author recommends the following iterative procedure, variations of which have been proposed, originally for the problem with $d_{ij} \geq 0$, by Ford [10], Moore [11], Bellman [12], and undoubtedly others. This procedure repeatedly updates all node labels (which, for the initial condition given in Eq. (1), represent, at iteration k, the lengths of the shortest paths connecting node 1 and the labelled nodes containing k+1 or less arcs) and none are considered final until all are. If $f_i^{(k)}$ represents the label of node i at iteration k, the fundamental recursion is

$$f_i^{(k+1)} = \min_j[d_{ji} + f_j^{(k)}] \qquad (1)$$

$$f_i^{(0)} = d_{1i} .$$

For the case $d_{ij} \geq 0$, convergence occurs whenever $f_i^{(k)} = f_i^{(k+1)}$ for all i, or after N-2 iterations if the former situation does not occur sooner (since no shortest path contains more than N-1 arcs). If (N-2) iterations are required, $(N-2)(N-1)^2$ additions and comparisons take place. The method is inefficient for the positive-distance problem if 2 or more iterations of equation (1) are needed; and, unless all N-2 iterations are required, as many iterations will be needed as the number of arcs in the shortest path from node 1 to node j, where node j is the node whose shortest path has the greatest number of arcs.*

*In a personal communication, C. Witzgall points out that for networks with far less than the allowable N(N-1) arcs, the Ford iterative procedure [10] may be superior to Dijkstra's. This is because the Dijkstra algorithm, even in this case, must check indices distinguishing permanent from temporary labels for N-1 nodes at each of N-1 iterations and hence requires at least $(N-1)^2$ comparisons

(Convergence may be enhanced, and cannot be slowed, by using $f_j^{(k+1)}$ on the right in Eq. (1) as soon as it has been computed. However, given any such immediate updating scheme, examples can be constructed where no improvement in rate of convergence results.) When the above procedure is applied to the problem with some negative d_{ij}, either convergence will occur for $k \le N-1$, indicating no negative cycles exist and the solution is optimal, or a change in some f_i will occur on the $(N-1)^{st}$ iteration, indicating the existence of a negative loop.

The above straightforward and easily programmed procedure appears slightly preferable to that given by Dantzig [13]. His scheme is unesthetic to prove, unstraightforward to program, and involves kN^3 $(0 < k < 1)$ additions and comparisons, even when the labelling of the set T (see his Theorem 4) is carried out in the efficient fashion of Dijkstra, described above in a different context. Dantzig's own bound, based on the admittedly misleading assumption of effortless sorting, is $N^3/2$ (his Eq. (17)).

Interestingly, the two procedures recommended below for solving the shortest route problem for all pairs of

no matter how few arcs are present; while one updating of all node labels by Ford's procedure involves only as many additions and comparisons as there are arcs.

nodes easily detect negative loops, if such exist for the problem with some $d_{ij} < 0$, and are in some cases more efficient for detecting negative loops than the scheme described above. However, they are never more efficient for establishing the optimal path between a particular pair of nodes if there are no negative loops.

III. THE SHORTEST PATHS BETWEEN ALL PAIRS OF NODES OF A NETWORK

Two somewhat different, but equally elegant and efficient, algorithms are recommended. One was published without comment as an obscure 9-line ALGOL algorithm in 1962 by Floyd [14] based on a theorem by Warshall [15] and was rediscovered and appropriately extolled in 1965 by Murchland [16]. The other was produced in 1966 by Dantzig [17]. Since both require exactly the same number of calculations--N(N-2)(N-2) additions and comparisons for the case $d_{ij} \geq 0$--are easily proved and programmed, and culminate a steady progression of successive improvements ([18]. [12], [19], [20]; actually [14] precedes the inferior algorithm of [20]), there is good reason to believe that they are definitive. While it is recommended that the reader consult the primary sources cited above, we shall give very brief descriptions.

The Floyd procedure builds optimal paths by inserting nodes, when appropriate, into more direct paths. Starting with the NxN matrix D of direct distances, N matrices are constructed sequentially. The k^{th} such matrix can be interpreted as giving the lengths of the shortest allowable paths between all node pairs (i,j) where only paths with intermediate nodes belonging to the set of nodes numbered from 1 through k are allowed. The (k+1)st matrix is constructed from the kth by using the formula

$$d_{ij}^{(k+1)} = \min \left[d_{ij}^{(k)},\ d_{i,k+1}^{(k)} + d_{k+1,j}^{(k)} \right] . \qquad (2)$$

$$d_{ij}^{(0)} = d_{ij} .$$

Here, k, which is initially zero, is incremented by 1 after i and j have ranged over the values 1,...,N and k = N-1 at termination. To see why this procedure is valid, suppose the shortest path from node 8 to node 5 is 8 - 3 - 7 - 1 - 9 - 5. Iteration 1 will replace d_{79} by $d_{71} + d_{19}$, iteration 3 will replace the current value of d_{87} (which may or may not be the original value) by $d_{83} + d_{37}$ (the optimal value) iteration 7 will replace the current d_{89} by $d_{87} + d_{79}$ (where these numbers are the optimal values as computed above), and iteration 9 will obtain for d_{85} the sum of d_{89} and d_{95} when d_{89} is as computed at iteration 7. Hence, the correct shortest distance is obtained. (The above indicates the proof but of course is not one.) A minor modification involving consideration of diagonal elements yields correct results when some d_{ij} are negative but no negative loops exist, as well as an indication of such if negative loops do exist. An additional advantage of this procedure is that N-1 additions and comparisons are easily circumvented whenever an element $d_{i,k+1}^{(k)}$ equals infinity in (2) (i.e., no path exists connecting nodes i and k+1 with only nodes 1 through k as intermediate nodes).* As in the case of the particular initial-terminal pair problem, an optimal policy table (matrix) associating with the initial-terminal node pair (i,j) the next node along the best path from i to j can be developed during the computation, or can be deduced from the final shortest-distance matrix.

Dantzig's scheme generates successive matrices of increasing size. At the k^{th} iteration a $k \times k$ matrix is produced whose elements are the lengths of the shortest paths connecting nodes i and j, $i=1,\ldots,k$, $j=1,\ldots,k$, which contain only nodes numbered between 1 and k as intermediate nodes. Given the $k \times k$ matrix $D^{(k)}$ with elements $d_{ij}^{(k)}$ as defined above, $D^{(k+1)}$ is computed as follows:

*The additional computation introduced in order to test for the presence of infinities adds about 5 percent to the computing time. For a sample problem with 10 nodes and 34 arcs and, hence, 56 infinities initially, the test yielded a 5 percent net improvement over no test. The amount of net improvement or degradation depends on the actual network configuration as well as the number of non-existent arcs.

1. Compute $d_{i,k+1}^{(k+1)}$ for $i=1,\ldots,k$ by

$$d_{i,k+1}^{(k+1)} = \min_{1\le j\le k}\left[d_{ij}^{(k)} + d_{j,k+1}\right]$$, and similarly compute $d_{k+1,i}$.

2. Compute $d_{ij}^{(k+1)}$ for $i=1,\ldots,k$, $j=1,\ldots,k$ by

$$d_{ij}^{(k+1)} = \min\left[d_{ij}^{(k)},\ d_{i,k+1}^{(k+1)} + d_{k+1,j}^{(k+1)}\right].$$

That the $d_{ij}^{(k)}$ yielded by the above steps are as previously defined is obvious after a little thought. (A proof is given in Ref. 17.) If all distances are positive, $d_{ii}^{(k)} = 0$ for all i and k. If not, d_{ii} can be easily computed and if any $d_{ii}^{(k)}$ is negative, a negative loop exists. It does not appear that the above algorithm can exploit non-existent arcs in a manner similar to Floyd's.

If the above algorithms, requiring N(N-1)(N-2) additions and comparisons, are indeed as efficient as possible, then the most efficient possible particular-pair algorithm must require at least (N-1)(N-2) such calculations (assuming, as was the case in Sec. II, that such procedures must--at least in the worst case--generate best paths from the initial node to all other nodes). The best algorithm of

Sec. II, that which required no elaborate data-preparation or list keeping, involves about $3N^2/2$ additions and comparisons. This gives, as a reasonable upper-bound estimate of potential future improvements for the particular-pair-of endpoints problem, a reduction in computation time of 33 percent. Computational experiments have confirmed that computing optimal paths between all pairs of nodes by means of N applications of the Dijkstra method requires one and one-half-times the time consumed by either of the above algorithms specifically solving the all-pairs problem. Viewed from the perspective of a combinatorialist, the well is rather dry. (Such is not the case for the N^3 procedure recommended above for the problem with negative distances.)

IV. DETERMINATION OF THE SECOND SHORTEST PATH

Sometimes it is desirable to know the value of the second (and third, etc.) shortest path through a network. For example, suppose that some complex quantitative (or even qualitative) feature characterizes paths, and the shortest path possessing this additional attribute is sought. By ignoring the special aspect in question and ordering paths from shortest to longest, the best path with the additional feature can sometimes be determined fairly efficiently.

In what follows, we shall initially restrict our attention to the problem of determining the second-best path between a specified initial node 1, and a specified destination, N. Then, conclusions will be drawn for more general problems. Two paths which do not visit precisely the same nodes in the same order are considered different. We exclude consideration of ties in our discussion by assuming, for simplicity, that all paths have different values. A path with a loop is considered an admissible path, and indeed such a path can be second best, even for problems with all $d_{ij} > 0$. Even node N may be visited twice along the second-best path.

The earliest good algorithm known to this author was proposed by Hoffman and Pavley [21]. A <u>deviation</u> from the shortest path was defined to be a path that coincides with the shortest path from its origin up to some node j on the path (j may be the origin or the terminal node), then deviates directly to some node k not the next node of the shortest path, and finally precedes from k to the fixed terminal node via the shortest path from k. The second shortest path between specified initial and terminal nodes is shown in Ref. 21 to be a deviation from the shortest path. To solve the problem posed above, first the shortest paths from all initial nodes to the specified destination are determined by any efficient algorithm. Then, all deviations from the shortest path between the specified origin and terminus are determined, evaluated, and compared, and the best noted. If the average node has M outgoing links, and the average shortest path contains K arcs, an average problem is solved in approximately MK additions and comparisons beyond those required for solution of the shortest path problem.

Suppose second-shortest paths from all nodes to the specified terminal node N are sought. Then the following modification of the Hoffman-Pavley method [21] seems appropriate. After solving the shortest-path problem, determine v_N, the length of the second-shortest path from N to N (it may be ∞), by considering all deviations at N plus best paths to N. Then, for each node k whose shortest path to N contains only one arc, compare (a) the length of the shortest path deviating at k and (b) $d_{kN} + v_N$. The minimum of these two quantities is v_k, the length of the second-shortest path from that node. Then, consider all nodes j which are two arcs from N via the shortest path. For each, compare (a) the length of the shortest path deviating at j and (b) the length of the arc out of j which is the first arc of the shortest path from j to N plus v evaluated at the terminal node of that arc. The minimum value is v_j. Repeat this iterative process until all nodes are labelled. Note that the iteration is performed on an index representing the number of arcs in the shortest path from each node. This procedure requires about MN additions and comparisons.

Reference 22, published subsequently to method 1 above, gives a seemingly different procedure. Define u_i as the length of the shortest path from node i to a specified terminal node N and v_i as the length of the second shortest path. Define $\min_k (x_1,\ldots,x_n)$ as the k^th smallest value of the quantities x_i. Then, according to Ref. 22, v_i is characterized by the equation

$$v_i = \min \begin{bmatrix} \min_2\limits_{j \neq i} (d_{ij} + u_j) \\ \min_1\limits_{j \neq i} (d_{ij} + v_j) \end{bmatrix} , \qquad i=1,\ldots,N-1$$

$$v_N = \min_1\limits_{i} \left[d_{Ni} + u_i \right] . \tag{3}$$

(The u_i are computed first, as in method 1, by any efficient procedure.)

The term $\min_2(d_{ij} + u_j)$ determines the value of the best path originating at node i and deviating from the shortest path at that node i. The term $\min_1(d_{ij} + v_j)$ determines the best path consisting of any first arc plus the second best continuation. What was not noted by the originators of the method or by Pollack [23] is that if the minimizing node in the $\min_1$ operation is not k (the next node of the already known shortest path from i) but some other node p, then $d_{ip} + u_p$ (an admissible solution to the $\min_2$ expression) is less than $d_{ip} + v_p$ (since $u_p < v_p$). Since the term $\min_1(d_{ij} + v_j)$ can only yield the overall minimum in Eq. (3) if $j = k$, where k is the node after i on the shortest path from i, the $\min_1$ term in Eq. (3) can be replaced by merely $d_{ik} + v_k$. After this reduction by a factor of 2 in required computation, we can calculate the method's approximate computational requirements. Bellman and Kalaba [22] recommend solution of Eq. (3) by an iterative procedure, where v_i is superscripted on the left by (k + 1) and v_j on the right by (k). Defining M and N as above and L as the average number of iterations until convergence of the iterative solution of Eq. (3), the method requires NML additions and comparisons on the average. L is less than N-1 and may be as large as the number of arcs in the shortest path containing the most arcs. However, after replacing the $\min_l$ term in Eq. (3) by $d_{ik} + v_k$ as discussed above, solution can be obtained by a one-pass scheme first labelling nodes one-arc-by-shortest-path from N, then two-arcs-by-shortest-path etc. This reduces the Bellman-Kalaba [22] procedure to precisely the modified Hoffman-Pavley [21] algorithm recommended above.

In summary, if only the second shortest path connecting a particular pair of nodes is desired, method 1 above is clearly the best since it requires MK calculations compared to MN for method 2, which [illegible]lve the all-initial-node problem in order to solve the particular-initial-node case. If problems involving all nodes as initial nodes and a fixed terminal node are to be solved, the methods as modified in this Memorandum are equivalent.

These conclusions strongly contradict those of Pollack's paper [23].

For determination of the third shortest paths from all initial nodes to N we recommend the following generalization of the above procedure. If w_i represents the length of the third shortest path from i, then

$$w_i = \min \begin{bmatrix} d_{ik} + w_k \\ \min_2 [d_{ij} + u_j] \\ j \neq i \end{bmatrix} \qquad (4)$$

if a single node k follows i along both the first and second shortest paths. If k is the node following i on the shortest path and m is the node following i on the second shortest, then

$$w_i = \min \begin{bmatrix} d_{ik} + v_k \\ d_{im} + v_m \\ \min_3 [d_{ij} + u_j] \\ j \neq i \end{bmatrix} . \qquad (5)$$

The function w_i can be computed node-by-node by first computing w at nodes that are one arc from N via the shortest path, then two arcs, etc. This one-pass procedure generalizes to the nth best path problem, because the same function appears on the left and right in the appropriate equation of the type of Eq. (4) above only if the pth best path from i for $p=1,\ldots,n-1$ all go to the same second node.

V TIME-DEPENDENT LENGTHS OF ARCS

At least one paper [24] has studied the problem of finding the fastest path between cities where the time of travel between city i and city j depends on the time of departure from city i. When t is the time of departure from city i for city j, let $d_{ij}(t)$ denote the travel time. (If travel schedules are such that a delay before departure decreases the time of arrival, $d_{ij}(t)$ represents the elapsed time between time t and the earliest possible time of arrival.) This model has applications in the areas of transportation planning and communication routing.* Cooke and Halsey [24] define $f_i(t)$ as the minimum time of travel to N, starting at city i at time t, and establish the formula

$$f_i(t) = \min_{j \neq i} \left[d_{ij}(t) + f_j(t + d_{ij}(t)) \right] \qquad (6)$$

$$f_N(t) = 0 .$$

*In more realistic communication routing problems, links are sometimes unavailable due to failure or overloading. Adaptive routing techniques that adjust to network changes are studied in Refs. 25 and 26.

Assuming all the $d_{ij}(t)$ are defined at, and take on, positive integer values, an iterative procedure is given for finding the quickest paths from all cities to city N, starting at city i at time 0. Defining T to be the maximum taken over all i of $d_{iN}(0)$ (a smaller T can be determined, at some inconvenience, if this number is infinite), and assuming all cities are connected at all times (perhaps by links taking infinite time), the procedure requires at most $N^2 T^2$ additions and comparisons.

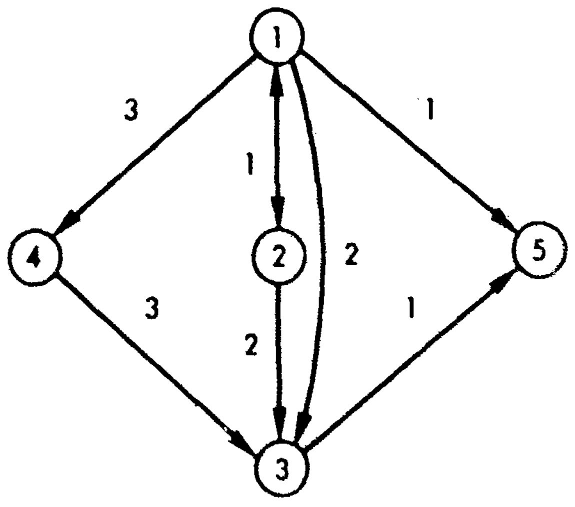
We wish to point out here that this problem can be solved by the method by Dijkstra [1] discussed in Sec. II above just as efficiently as can the problem where the times (or distances) are not time-dependent. Also, the restriction to integer-valued times can be dropped and any real-valued times can be treated. Define the tentative node (city) label f_i to be an upper bound on the earliest time of arrival at node i, and permanent labels to be earliest possible (optimal) times-of-arrival. First, permanently label node i_o (the initial node) zero and all other nodes infinity. Next, tentatively label all nodes j with the minimum of the current node label f_j and the sum of f_{i_o} and $d_{i_o j}(f_{i_o})$. Then, find the minimum, non-permanent node label, say f_k, and declare it permanent.

(f_k is the earliest possible time of arrival at node k, leaving node i_o at time 0.) Node k is then used to possibly reduce the labels at all tentatively labelled cities, by comparing $f_k + d_{kj}(f_k)$ to the current label, and the minimum new temporary label is made permanent, etc. After at most N^2 comparisons and $\frac{N^2}{2}$ additions, city N is labelled, and the quickest paths, leaving i_o at time 0, to all nodes, including N, are determined. As is the case for the closely related Dijkstra procedure [1], about $3N^2/2$ additions and comparisons are required when a method of distinguishing temporarily from permanently-labelled nodes is implemented. If quickest paths from all cities to N are desired, the algorithm must be repeated N-1 times; but, even then, the procedure compares quite favorably, with respect to both required assumptions and computation, with the Cooke-Halsey algorithm [24].

VI. SHORTEST PATHS VISITING SPECIFIED NODES

Given a set of N nodes and distances $d_{ij} \geq 0$, suppose it is desired to find the shortest path between nodes 1 and N which passes through the k-1 nodes 2, 3,...,k $\leq$ N-1, called "specified nodes." A simple, but completely erroneous, solution of this problem was reported by Saksena and Kumar [27]. Noting this, we wish to give a solution method.

The fallacy in Ref. 27 is the assertion (subject to a proviso to follow) that the shortest path from a specified node i to N passing through at least p of the specified nodes enroute is composed of the shortest path from i to some specified node j followed by the shortest path from j to N passing through at least p-r-1 specified nodes, where r is the number of specified nodes that turn out to lie on the shortest unrestricted path from i to j. Saksena and Kumar [27] erroneously assert that this is true provided--should specified nodes occurring on the shortest path from i to j also lie on the continuation path from j to N and therefore be counted twice--that at least p distinct specified nodes lie on the path. Should some candidate path corresponding to some j violate the duplication of nodes proviso above, that possibility is inadmissible and going initially from i to j is dropped from consideration. The procedure fails to note that, in this case, some less short continuation from j passing through at least p-r-1 specified nodes and avoiding duplication of nodes may yield a better path than the best remaining path satisfying the conditions described above. For example, in the network shown below, with all nodes considered specified, suppose we seek the best path from 1 to 5 passing through at least two intermediate nodes.



The best path from 1 to 4 has length 3 and passes through no nodes enroute, and the best from 4 to 5 passing through at least one node has length 4; hence, this possibility has length 7. The best path from 1 to 3 and best continuation from 3 passing through one node has length infinity (no such continuation from 3 exists). The best path from 1 to 2 has length 1 and the best continuation, passing through one node enroute, from 2 to 5 has length 2 (it returns to node 1) yielding a sum of 3. However, this possibility is eliminated by the fact that node 1 is counted twice. The answer, by the Saksena-Kumar algorithm [27] would then be 7, the best of the other alternatives. Yet, the path 1-2-3-5 has length 4 and is admissible. This is an example of a best first portion and a second-best continuation being optimal. (Or, the same path can alternatively be viewed as the second-best path from 1 to 3 followed by the best continuation.) No simple modification of the referenced method seems to handle this kind of situation.

Assuming paths with loops are admissible, the problem can be correctly solved as follows. First solve the shortest path problem for the N-node network for all pairs of initial and final nodes. Let d'_{ij} represent the length of the shortest path from node i to j. Then, solve the (k + 1)-city "traveling-salesman" problem for the shortest path from 1 to N passing through nodes 2, 3,...,k where the distance from node i to j is d'_{ij}. Methods of solution are discussed by Gomory [28]. While there are no easy solutions for the traveling-salesman problem, it is hard to see how the specified-city problem can possibly be any easier than the traveling-salesman problem of dimension $k + 1$ since if $k = N-1$ it <u>is</u> the traveling-salesman problem.

VII. CONCLUSION

We have given some observations and recommendations for computationally treating discrete shortest-path problems of varying degrees of generality. Our intent has not been definitive solution, but rather to clear the air by presenting both some important methods and references and some critical comments and warnings.

REFERENCES

1. Dijkstra, E. W., "A Note on Two Problems in Connexion with Graphs," Numerische Mathematik, 1, 1959, pp. 269-271.
2. Whiting, P. D., and J. A. Hillier, "A Method for Finding the Shortest Route through a Road Network," Operational Research Quarterly, Vol. 11, Nos. 1/2, March/June 1960, pp. 37-40.
3. Dantzig, G. B., "On the Shortest Route through a Network," Management Science, Vol. 6, No. 2, January 1960, pp. 187-190.
4. Pollack, M., and W. Wiebenson, "Solution of the Shortest-Route Problem--A Review," Op. Res., Vol. 8, No. 2, March-April 1960, pp. 224-230.
5. Ford, L. R., Jr., and D. R. Fulkerson, "Constructing Maximal Dynamic Flows from Static Flows," Op. Res., Vol. 6, No. 3, May-June 1958, pp. 419-433.
6. Dantzig, G. B., Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey, 1963.
7. Berge, C., and A. Ghouila-Houri, Programming, Games and Transportation Networks, translated by Merrington and Ramanujacharyulu, Methuen, London (1965), p. 176.
8. Nicholson, T.A.J., "Finding the Shortest Route Between Two Points in a Network," The Computer Journal, Vol. 9, No. 3, November 1966, pp. 275-280.
9. Murchland, J. D., The "Once-Through" Method of Finding all Shortest Distances in a Graph from a Single Origin, Transport Network Theory Unit, London Graduate School of Business Studies, Report LBS-TNT-56, August 1967.
10. Ford, L. R., Jr., Network Flow Theory, The RAND Corporation, P-923, August 1956.
11. Moore, E. F., "The Shortest Path through a Maze," Proc. Int. Symp. on the Theory of Switching, Part II, Ap 2-5, 1957, The Annals of the Computation Laboratory of Harvard University 30, Harvard University Press, 1959.

12. Bellman, R. E., "On a Routing Problem," Quart. Appl. Math., Vol. XVI, No. 1, 1958.

13. Dantzig, G. B., W. O. Blattner and M. R. Rao, All Shortest Routes from a Fixed Origin in a Graph, Operations Research House, Stanford University, Technical Report No. 66-2, November 1966.

14. Floyd, R. W., "Algorithm 97, Shortest Path," Communications of the ACM, Vol. 5, No. 6, June 1962, p. 345.

15. Warshall, S., "A Theorem on Boolean Matrices," Journal of the ACM, Vol. 9, 1962, pp. 11-12.

16. Murchland, J. D., A New Method for Finding all Elementary Paths in a Complete Directed Graph, Transport Network Theory Unit, London School of Economics, Report LSE-TNT-22, October 1965.

17. Dantzig, G. B., All Shortest Routes in a Graph, Operations Research House, Stanford University, Technical Report No. 66-3, November 1966.

18. Shimbal, A., "Structure in Communication Nets," Proc. of the Symposium on Information Networks, Polytechnic Institute of Brooklyn, Ap. 12-14, 1954.

19. Investigation of Model Techniques, Second Annual Report, July 1957-1958, Case In[illegible]. of Tech., Cleveland, Ohio, ASTIA Report No. AD211968.

20. Hu, T. C., "Revised Matrix Algorithms for Shortest Paths," SIAM J. on Appl. Math., Vol. 15, No. 1, January 1967, pp. 207-218.

21. Hoffman, W., and R. Pavley, "A Method for the Solution of the Nth Best Path Problem," J. Assoc. Comp. Mach., Vol. 6, No. 4, October 1959, pp. 506-514.

22. Bellman, R., and R. Kalaba, "On kth Best Policies," J. SIAM, Vol. 8, No. 4, December 1960, pp. 582-588.

23. Pollack, M., "Solutions of the kth Best Route through a Network--A Review," J. Math. Anal. and Appl., Vol. 3, August-December 1961, pp. 547-559.

24. Cooke, K. L., and E. Halsey, "The Shortest Route through a Network with Time-Dependent Internodal Transit Times," J. Math. Anal. and Appl., Vol. 14, No. 3, June 1966, pp. 493-498.

25. Butrimenko, A. V., "On Searching for the Shortest Paths Along a Graph During Variations in It," Engineering Cybernetics, No. 6, Nov.-Dec. 1964, pp. 50-53 (translated from Russian).

26. Boehm, B. W., and R. L. Mobley, Adaptive Routing Techniques for Distributed Communications Systems, The RAND Corporation, RM-4781-PR, February 1966.

27. Saksena, J. P., and S. Kumar, "The Routing Problem with 'K' Specified Nodes," Op. Res., Vol. 14, No. 5, September-October 1966, pp. 909-913.

28. Gomory, R. E., The Traveling Salesman Problem, Proceedings IBM Scientific Computing Symposium on Combinatorial Problems, March 1964, pp. 93-117.

DOCUMENT CONTROL DATA

1. ORIGINATING ACTIVITY THE RAND CORPORATION	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED
	2b. GROUP
3. REPORT TITLE AN APPRAISAL OF SOME SHORTEST PATH ALGORITHMS	
4. AUTHOR(S) (Last name, first name, initial) Dreyfus, S. E.	

5. REPORT DATE October 1967	6a. TOTAL No. OF PAGES 42	6b. No. OF REFS. 28
7. CONTRACT OR GRANT No. F44620-67-C-0045	8. ORIGINATOR'S REPORT No. RM-5433-PR	
9a. AVAILABILITY/LIMITATION NOTICES DDC-1	9b. SPONSORING AGENCY United States Air Force Project RAND	

10. ABSTRACT

A critical review of the literature of shortest paths in networks that examines methods for determining (1) the shortest path between two specified nodes; (2) the shortest path between all pairs of nodes; (3) the second, third, etc., shortest path; (4) the fastest path through a network with travel times depending on the departure time; and (5) the shortest path between specified endpoints that passes through specified intermediate nodes. Inefficient algorithms, erroneous procedures, and false assumptions in the current literature are identified.

11. KEY WORDS

Mathematics
Network theory